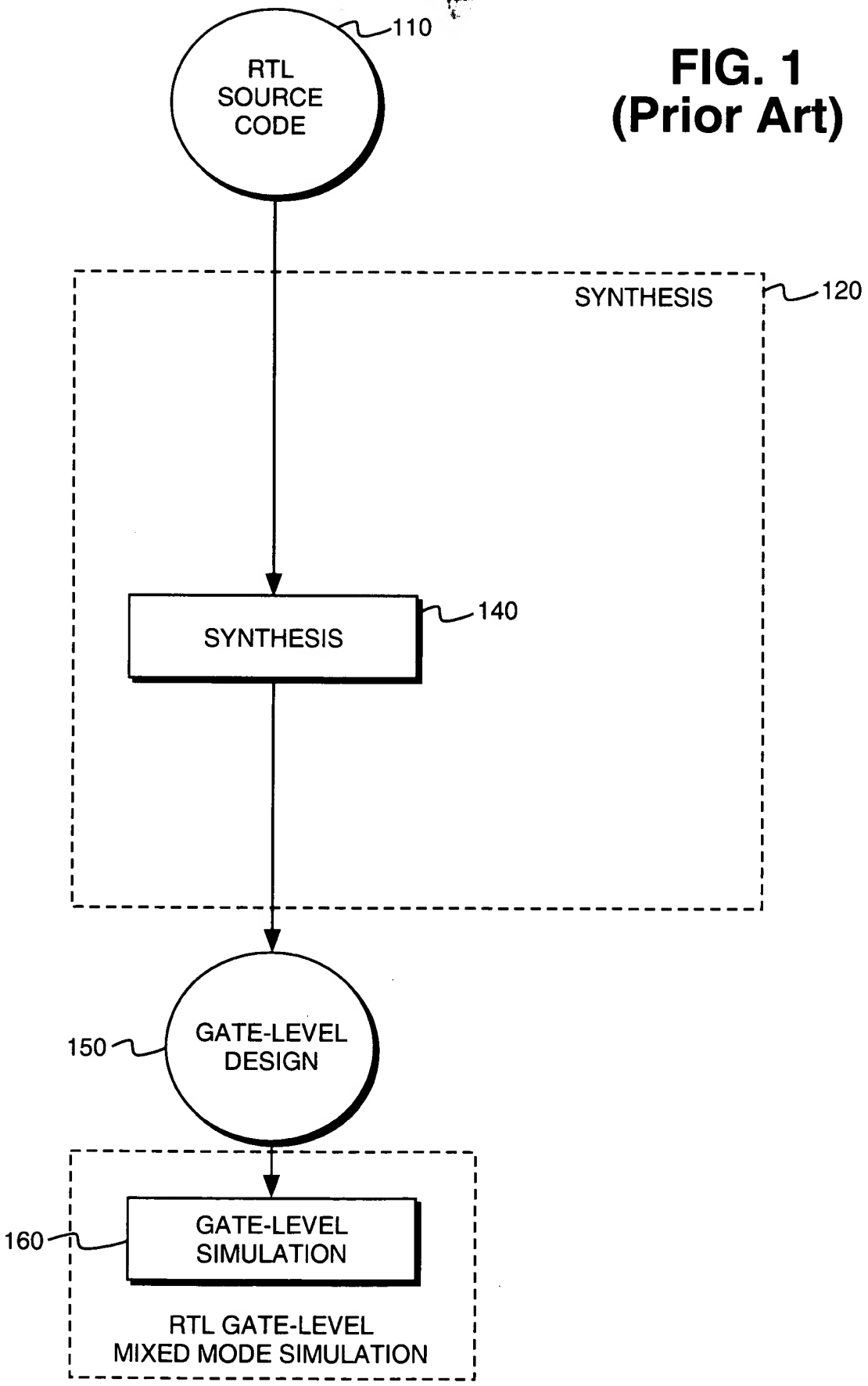


FIG. 1
(Prior Art)

FIG. 2

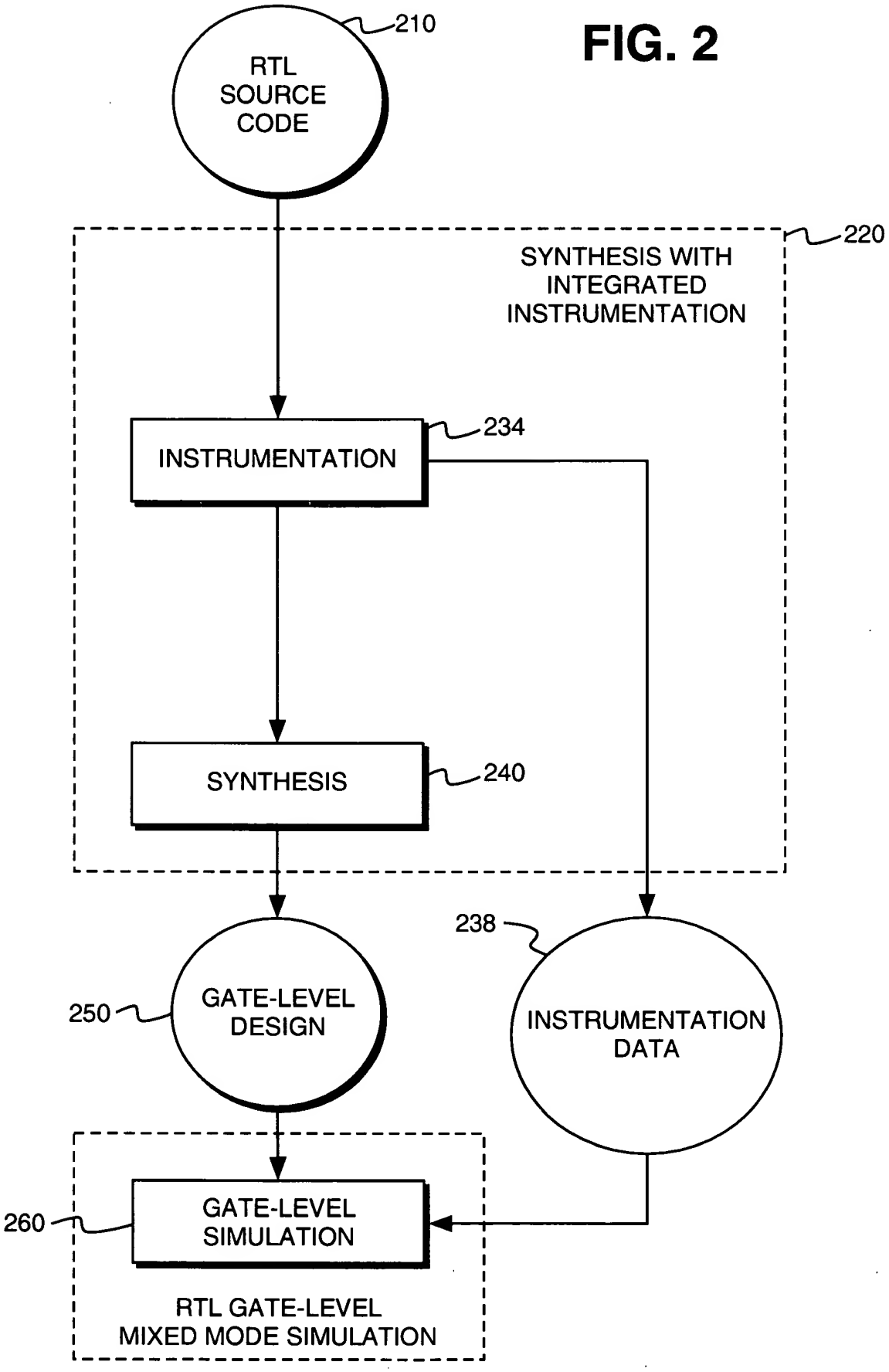

RTL SOURCE CODE
210
SYNTHESIS WITH INTEGRATED INSTRUMENTATION
220
INSTRUMENTATION
234
SYNTHESIS
240
238
INSTRUMENTATION DATA
250
GATE-LEVEL DESIGN
260
GATE-LEVEL SIMULATION
RTL GATE-LEVEL MIXED MODE SIMULATION

FIG. 3

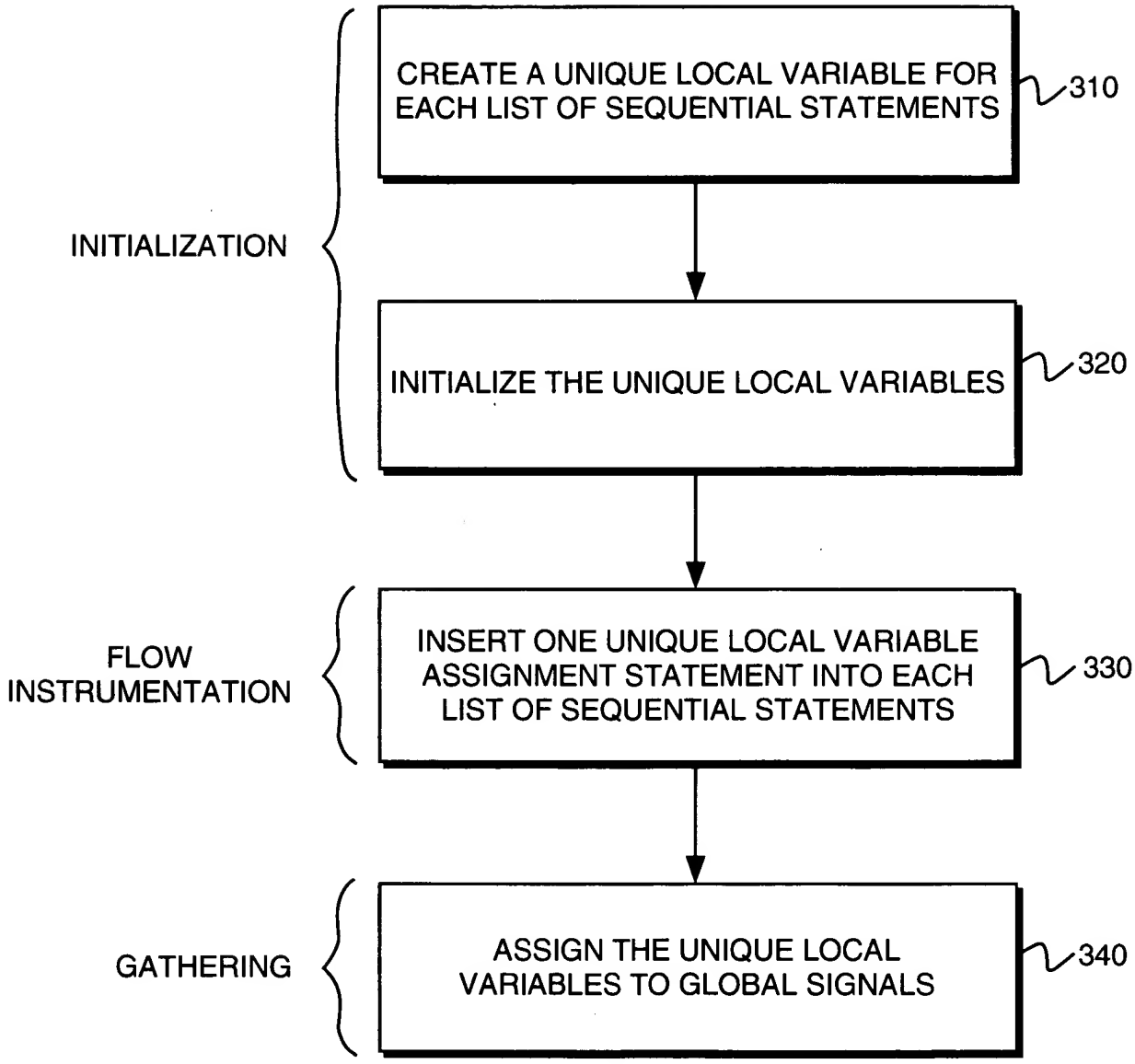

CREATE A UNIQUE LOCAL VARIABLE FOR EACH LIST OF SEQUENTIAL STATEMENTS
310
INITIALIZATION
INITIALIZE THE UNIQUE LOCAL VARIABLES
320
FLOW INSTRUMENTATION
INSERT ONE UNIQUE LOCAL VARIABLE ASSIGNMENT STATEMENT INTO EACH LIST OF SEQUENTIAL STATEMENTS
330
GATHERING
ASSIGN THE UNIQUE LOCAL VARIABLES TO GLOBAL SIGNALS
340

FIG. 4

400

```
ENTITY ALOOP IS

PORT(
        A : IN BIT_ VECTOR ( 0 TO 1 ) ;
        RESET : IN BOOLEAN;
        STATUS : OUT BOOLEAN ) ;

END ENTITY ALOOP ;


ARCHITECTURE RTL OF ALOOP IS

BEGIN

PROCESS ( A, RESET )

VARIABLE ZEROS, ONES : INTEGER ;

BEGIN
410 ---> IF ( RESET )                                  -- STATEMENT # 1
         THEN
420 --->     STATUS <= 0 ;                             -- STATEMENT # 2
         ELSE
430 --->     ZEROS := 0;                               -- STATEMENT # 3
440 --->     ONES := 0;                                -- STATEMENT # 4
450 --->     FOR I IN 0 TO 1 LOOP                      -- STATEMENT # 5
460 --->         IF A ( I ) = '0'                      -- STATEMENT # 6
                 THEN
470 --->             ZEROS := ZEROS + 1 ;              -- STATEMENT # 7
                 ELSE
480 --->             ONES := ONES + 1 ;                -- STATEMENT # 8
                 END IF ;

             END LOOP ;
490 --->     STATUS <= ( ZEROS > ONES ) ;              -- STATEMENT # 9
         END IF ;

END PROCESS ;
END ARCHITECTURE ;
```

FIG. 5

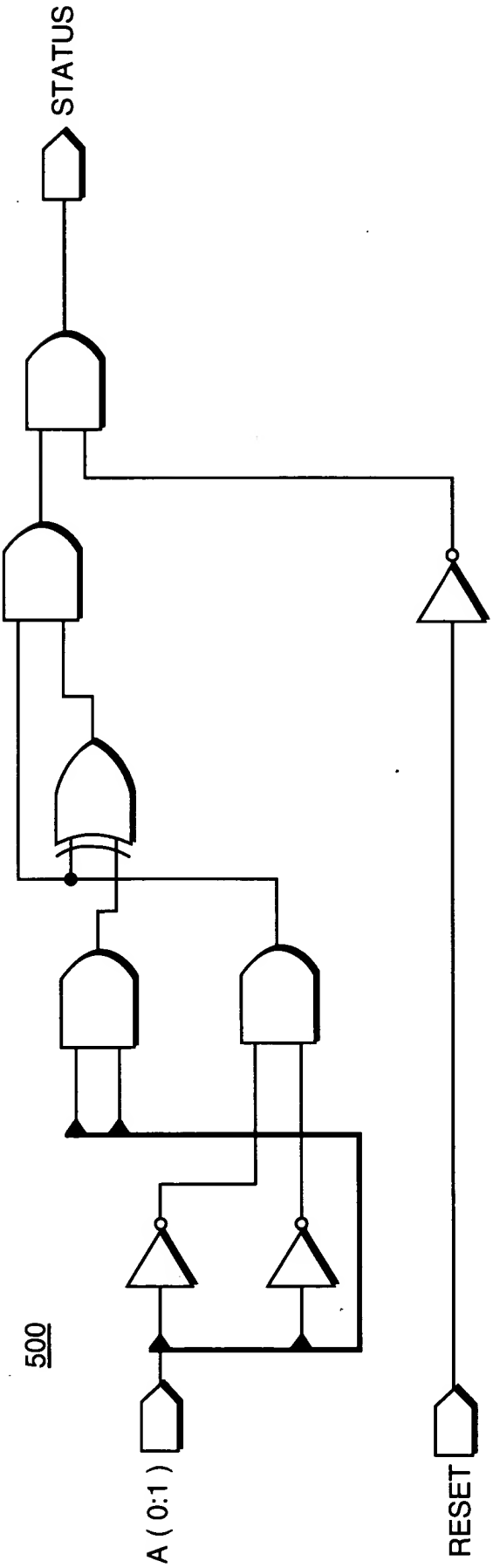
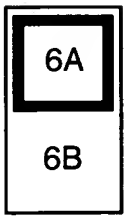


FIG. 6A



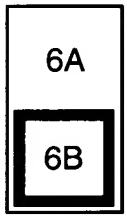
600

```
ENTITY ALOOP IS
PORT(
A : IN BIT_VECTOR (0 TO 1) ;
RESET : IN BOOLEAN ;
STATUS : OUT BOOLEAN ;
SIG_ TRACE1, SIG_ TRACE2, SIG_ TRACE3, SIG_ TRACE4, SIG_TRACE5,   } ~610
SIG_TRACE6 : OUT BIT
);
END ENTITY ALOOP ;

ARCHITECTURE RTL OF ALOOP IS
BEGIN
PROCESS (A, RESET)
VARIABLE TRACE1, TRACE2, TRACE3, TRACE4, TRACE5, TRACE6 : BIT ;   } ~612
VARIABLE ZEROS, ONES : INTEGER ;

BEGIN
TRACE1 :='0' ; TRACE2 :='0' ;
TRACE3 :='0' ; TRACE4 :='0' ;   } ~620
TRACE5 :='0' ; TRACE6 :='0' ;
```

FIG. 6B



600

```
630 ➤ TRACE1 := '1' ;                    -- INSTRUMENT STATEMENT #1
      IF (RESET)                         -- STATEMENT #1
      THEN
632 ➤    TRACE2 := '1' ;                 -- INSTRUMENT STATEMENT #2
         STATUS <= FALSE ;               -- STATEMENT #2
      ELSE
634 ➤    TRACE3 := '1' ;                 -- INSTRUMENT STATEMENT #3, #4, #5, #9
         ZEROS := 0 ;                    -- STATEMENT #3
         ONES := 0 ;                     -- STATEMENT #4
         FOR I IN 0 TO 1 LOOP            -- STATEMENT #5
636 ➤       TRACE4 := '1' ;              -- INSTRUMENT STATEMENT #6
            IF A (I) ='0' ;              -- STATEMENT #6
            THEN
638 ➤          TRACE5 := '1' ;           -- INSTRUMENT STATEMENT #7
               ZEROS := ZEROS +1 ;       -- STATEMENT #7
            ELSE
640 ➤          TRACE6 := '1' ;           -- INSTRUMENT STATEMENT #8
               ONES := ONES +1;          -- STATEMENT  #8
            END IF;
         END LOOP;

642 ➤ STATUS <= (ZEROS > ONES) ;         -- STATEMENT #9

      END IF ;
      SIG_ TRACE1 <= TRACE1 ; SIG_ TRACE2 <= TRACE2 ;
      SIG_ TRACE3 <= TRACE3 ; SIG_ TRACE4 <= TRACE4 ;   } ~ 650
      SIG_ TRACE5 <= TRACE5 ; SIG_ TRACE6 <= TRACE6 ;
      END PROCESS ;

      END ARCHITECTURE ;
```

FIG. 7

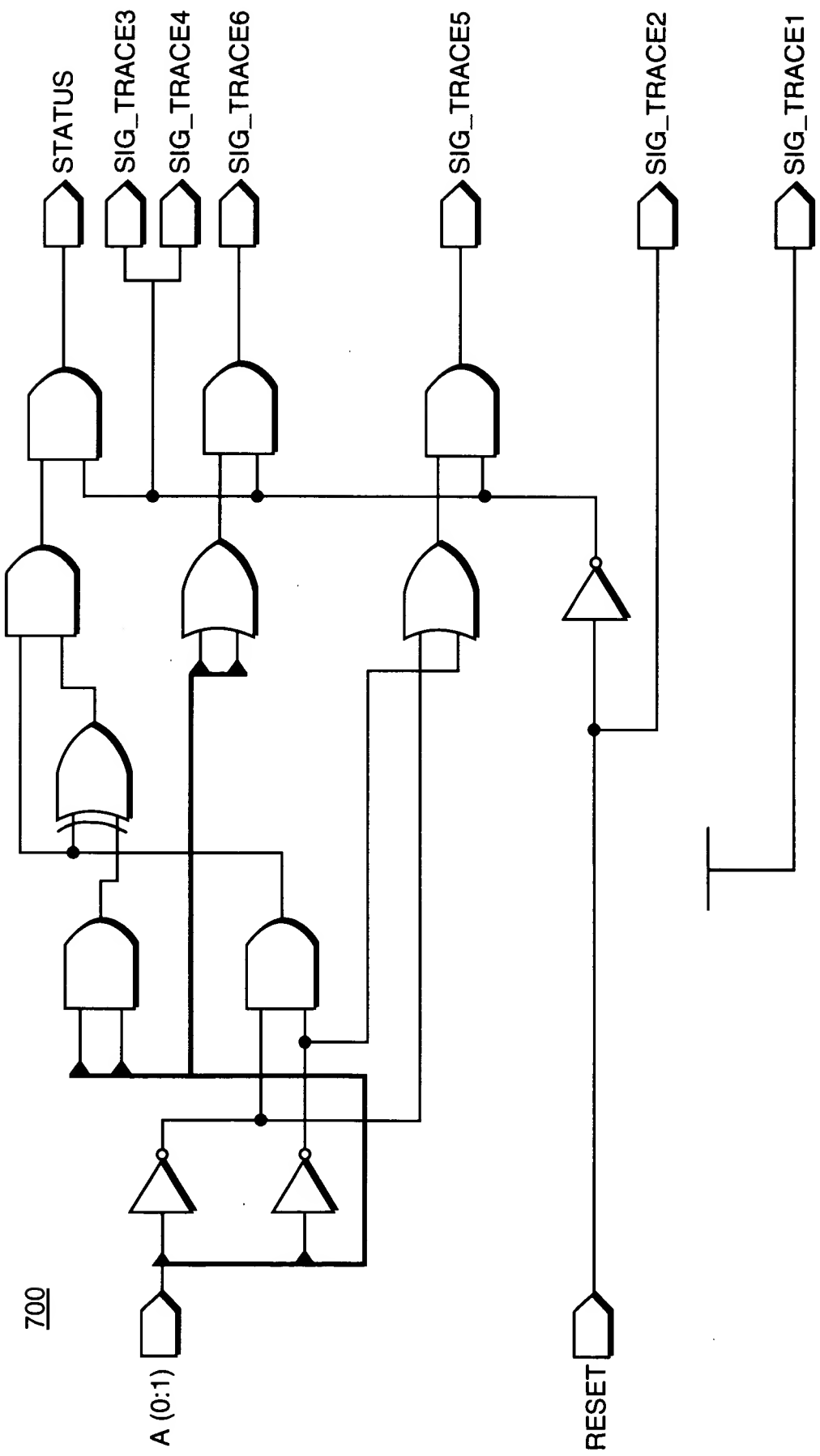

STATUS
SIG_TRACE3
SIG_TRACE4
SIG_TRACE6
SIG_TRACE5
SIG_TRACE2
SIG_TRACE1
700
A (0:1)
RESET

FIG. 8

800

```
MODULE SAMPLE (RESET, D, CLK, Q) ;

INPUT RESET ;
INPUT D ;
INPUT CLK ;
REG Q ;
OUTPUT Q ;


ALWAYS @ (CLK OR RESET OR D)
BEGIN
        IF (RESET==1)
            Q <= 0 ;
        ELSE
            IF (CLK==1)
                Q <= D ;
END


ENDMODULE
```

FIG. 9

900

```
MODULE SAMPLE
(RESET, D, CLK, Q, SIG_TRACE1, SIG_TRACE2, SIG_TRACE3, SIG_ TRACE4 ) ;

INPUT RESET ;
INPUT D ;
INPUT CLK ;
REG Q ;
OUTPUT Q ;

REG SIG_TRACE1, SIG_TRACE2, SIG_TRACE3, SIG_TRACE4 ;
OUTPUT SIG_TRACE1, SIG_TRACE2, SIG_TRACE3, SIG_TRACE4 ;

INTEGER TRACE1, TRACE2, TRACE3, TRACE4 ;

ALWAYS @ (CLK OR RESET OR D)
BEGIN
        TRACE1 = 0 ; TRACE2 = 0 ; TRACE3 = 0 ; TRACE4 = 0 ;
        TRACE1 = 1 ;
        IF (RESET==1)
        BEGIN
                TRACE2 = 1 ;
                Q <= 0 ;
        END
        ELSE
        BEGIN
                TRACE3 = 1 ;
                IF (CLK== 1)
                BEGIN
                        TRACE4 = 1 ;
                          Q <= D ;
                END
        END

SIG_TRACE1 = TRACE1 ;
SIG_TRACE2 = TRACE2 ;
SIG_TRACE3 = TRACE3 ;
SIG_TRACE4 = TRACE4 ;

END

ENDMODULE
```

FIG. 10

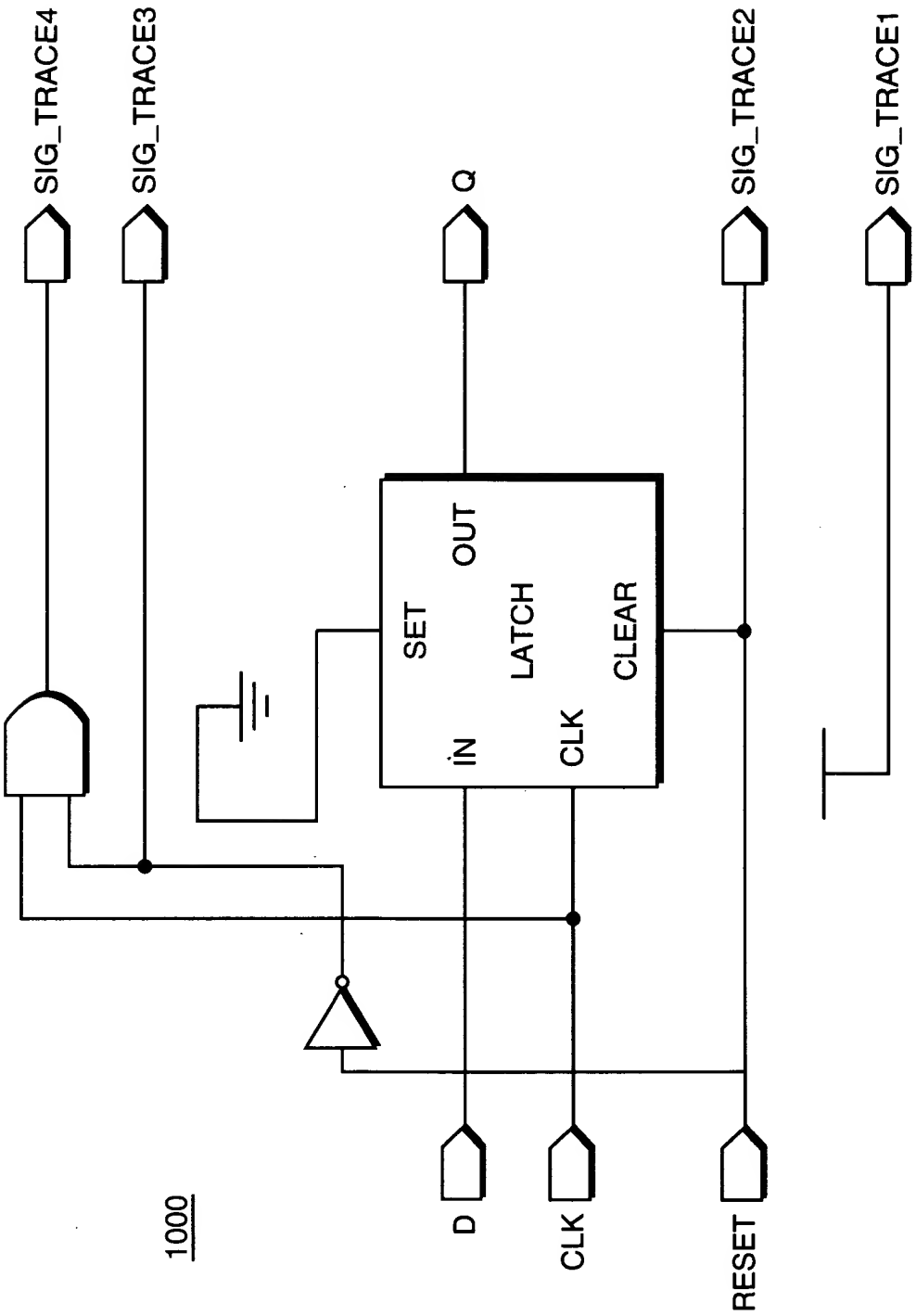

SIG_TRACE4
SIG_TRACE3
Q
SIG_TRACE2
SIG_TRACE1
SET
OUT
LATCH
CLEAR
IN
CLK
1000
D
CLK
RESET

FIG. 11

1100

```
PROCESS (CLK, D, RESET)
BEGIN
        IF (RESET = '1') THEN
                Q <= '0' ;

        ELSIF (CLK'EVENT AND CLK = '1') THEN
                Q <= D ;

        END IF;

END PROCESS
```

FIG. 12

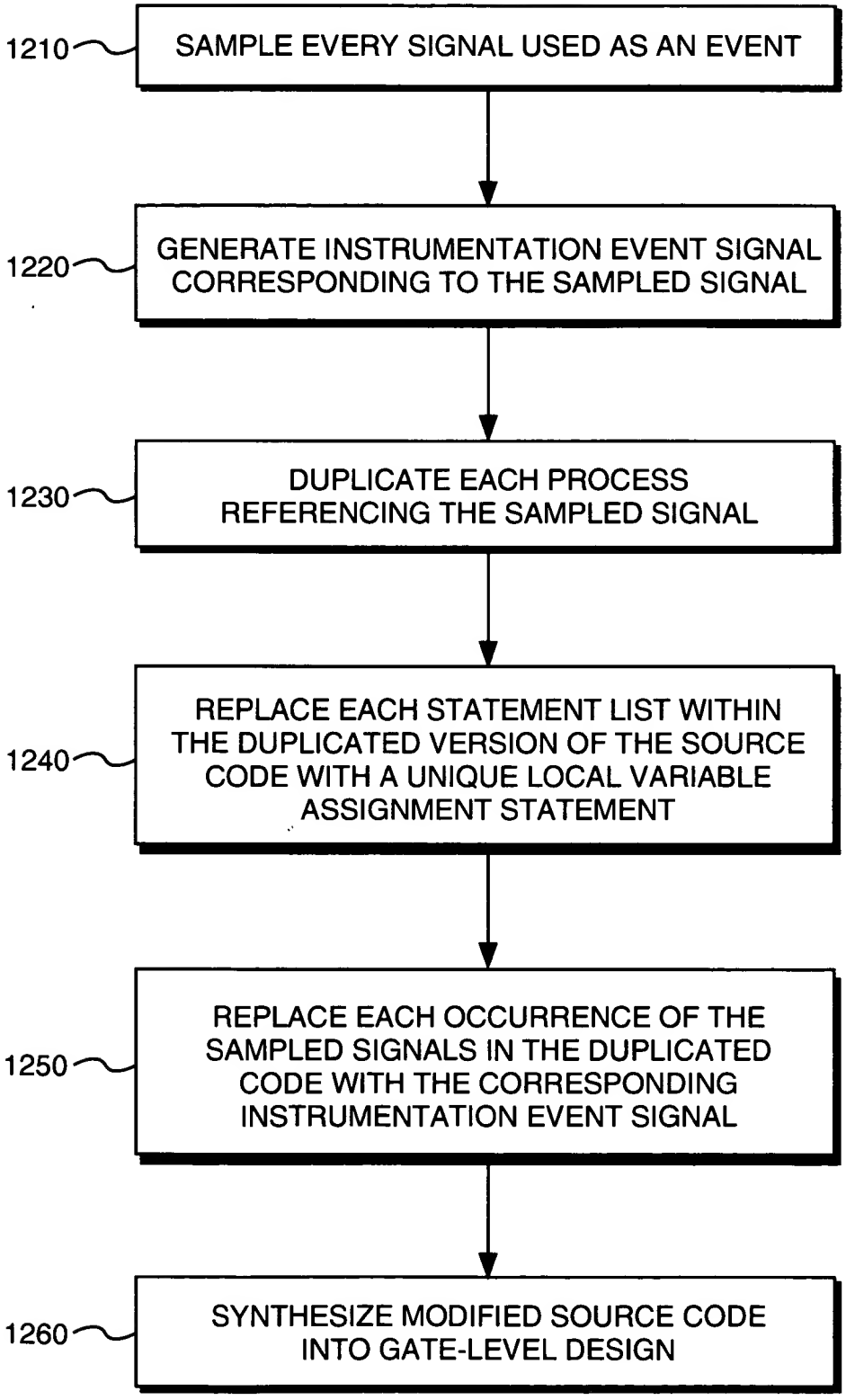

1210
SAMPLE EVERY SIGNAL USED AS AN EVENT
1220
GENERATE INSTRUMENTATION EVENT SIGNAL CORRESPONDING TO THE SAMPLED SIGNAL
1230
DUPLICATE EACH PROCESS REFERENCING THE SAMPLED SIGNAL
1240
REPLACE EACH STATEMENT LIST WITHIN THE DUPLICATED VERSION OF THE SOURCE CODE WITH A UNIQUE LOCAL VARIABLE ASSIGNMENT STATEMENT
1250
REPLACE EACH OCCURRENCE OF THE SAMPLED SIGNALS IN THE DUPLICATED CODE WITH THE CORRESPONDING INSTRUMENTATION EVENT SIGNAL
1260
SYNTHESIZE MODIFIED SOURCE CODE INTO GATE-LEVEL DESIGN

FIG. 13

1300

```
PROCESS (FAST_CLK)
BEGIN
        IF (FAST_CLK'EVENT AND FAST_CLK = '1')
        THEN
                SAMPLED_CLK <= CLK ;
        END IF
END PROCESS ;

CLK_EVENT <= SAMPLED_CLK / = CLK ;
CLK_STABLE <= SAMPLED_CLK = CLK ;
CLK_LASTVALUE <= SAMPLED_CLK ;
```

1310

```
PROCESS (CLK, D, RESET, CLK _ EVENT)
        VARIABLE TRACE1, TRACE2 : BIT ;
BEGIN
        TRACE1 := '0' ; TRACE2 := '0' ;
        IF (RESET = '1') THEN
                TRACE1 := '1' ;
        ELSIF (CLK_EVENT AND CLK = '1') THEN
                TRACE2 := '1' ;
        END IF;
SIG_TRACE1 <= TRACE1 ;  SIG_TRACE2 <= TRACE2 ;
END PROCESS ;
```

1320

```
PROCESS (CLK, D, RESET)
BEGIN
        IF (RESET = '1') THEN
                Q <= '0' ;
        ELSIF (CLK'EVENT AND CLK = '1') THEN
                Q <= D ;
        END IF ;
END PROCESS
```

1330

FIG. 14

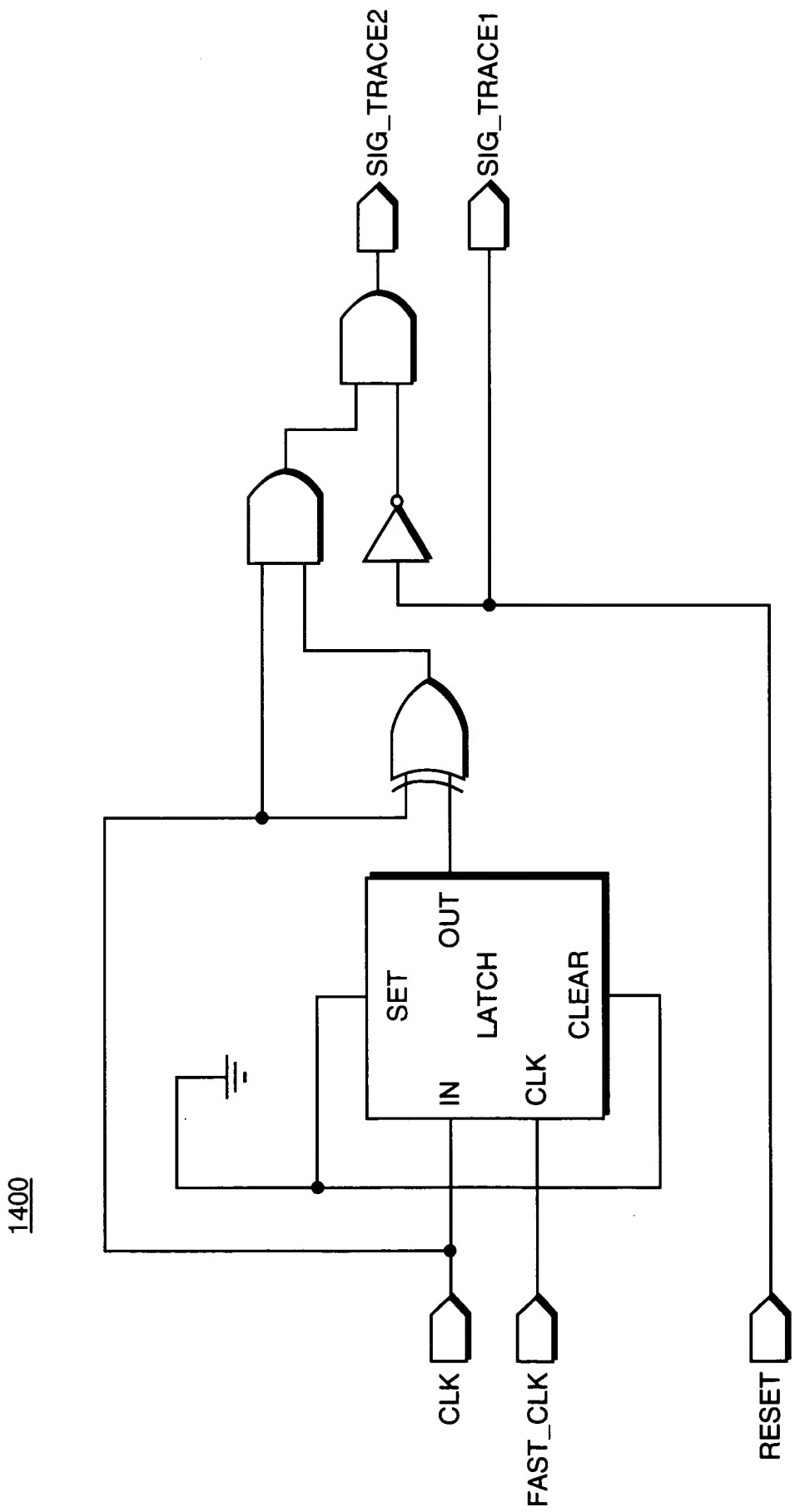

1400
SIG_TRACE2
SIG_TRACE1
SET
OUT
LATCH
CLEAR
IN
CLK
CLK
FAST_CLK
RESET

FIG. 15

1500

```
ALWAYS @ (POSEDGE CLK OR NEGEDGE RESET)
BEGIN
        IF (RESET == 0)
                Q <= 0 ;
        ELSE
                Q <= D ;
END
```

FIG. 16

1600

```
ALWAYS @ (POSEDGE FAST_CLK)
BEGIN
        SAMPLED_CLK <= CLK
        SAMPLED_RESET <= RESET ;
END
ASSIGN CLK_EDGE = SAMPLED_CLK ^ CLK ;
ASSIGN RESET_ EDGE = SAMPLED_ RESET ^ RESET;

INTEGER TRACE1, TRACE2;
REG [1:0] SIG_ TRACE ;
ALWAYS @ (CLK_EDGE OR RESET_EDGE OR CLK OR RESET)
BEGIN
        TRACE1 = 0 ; TRACE2 = 0 ;
        IF ((CLK_EDGE == 1) && (CLK == 1)II(RESET_EDGE == 1) && (RESET == 0))
                IF (RESET == 0)
                        TRACE1 =1;
                ELSE
                        TRACE2 =1;
                SIG_TRACE[0] = TRACE1 ;
                SIG_TRACE[1] = TRACE2 ;
END

ALWAYS @ (POSEDGE CLK OR NEGEDGE RESET)
BEGIN


        IF (RESET == 0)
            Q <=0 ;
         ELSE
            Q <= D ;
END
```

FIG. 17

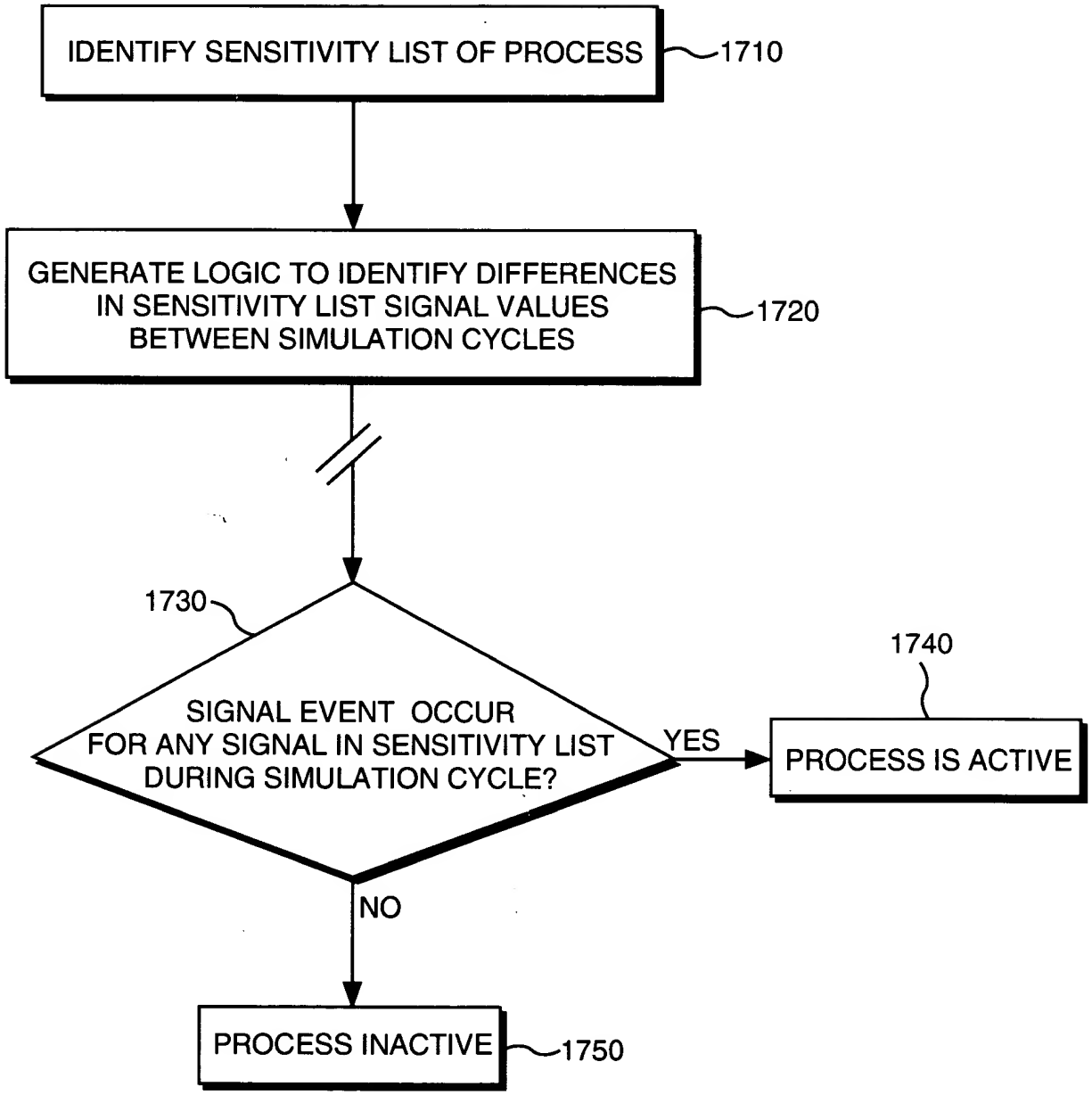

IDENTIFY SENSITIVITY LIST OF PROCESS
1710
GENERATE LOGIC TO IDENTIFY DIFFERENCES IN SENSITIVITY LIST SIGNAL VALUES BETWEEN SIMULATION CYCLES
1720
1730
SIGNAL EVENT OCCUR FOR ANY SIGNAL IN SENSITIVITY LIST DURING SIMULATION CYCLE?
YES
1740
PROCESS IS ACTIVE
NO
PROCESS INACTIVE
1750

FIG. 18

```
P1: PROCESS (A,B,C)

PROCESS (FAST_CLK)
 BEGIN
        IF (FAST_ CLK'EVENT AND FAST_ CLK='1')
         THEN
                SAMPLED_A <=A ;
                SAMPLED_B <=B ;
                SAMPLED_C <=C ;
        END IF
        END PROCESS;                                        } 1810

P1_ ACTIVE <= (SAMPLED_A /= A)
      OR (SAMPLED_B /= B)
      OR (SAMPLED_C /= C);                                  } 1820
```

FIG. 19

```
CASE OPCODE IS
    WHEN "00" => TRACE1 := 1;
                 STATE := 1;
    WHEN "01" => TRACE2 := 1;
                 STATE:= 2 ;
    WHEN "10" => TRACE3 := 1;
                 STATE := 2 ;
    WHEN "11" => TRACE4 := 1;
                 STATE := 1 ;
END CASE ;
```

1910

FIG. 20

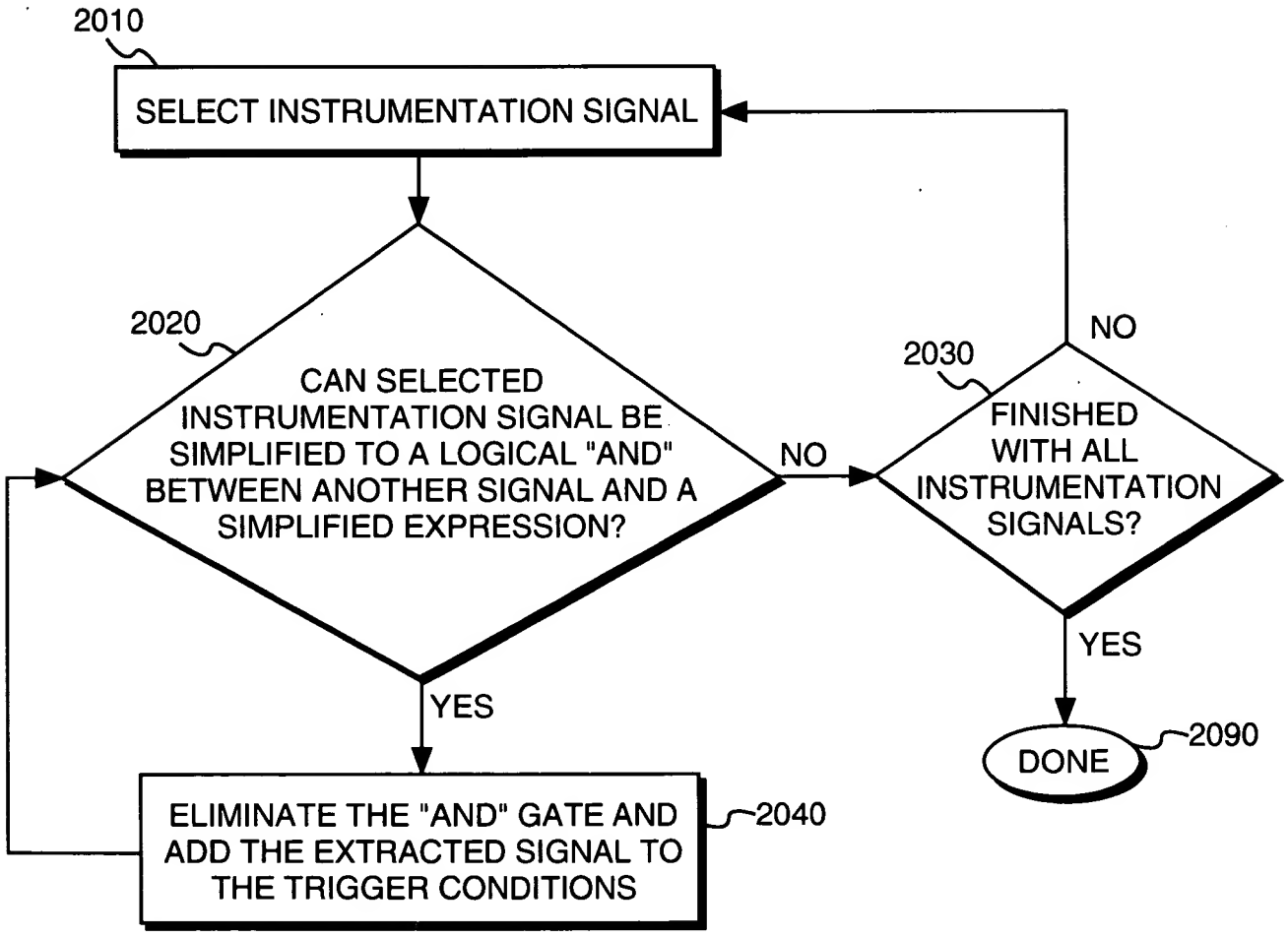


FIG. 21

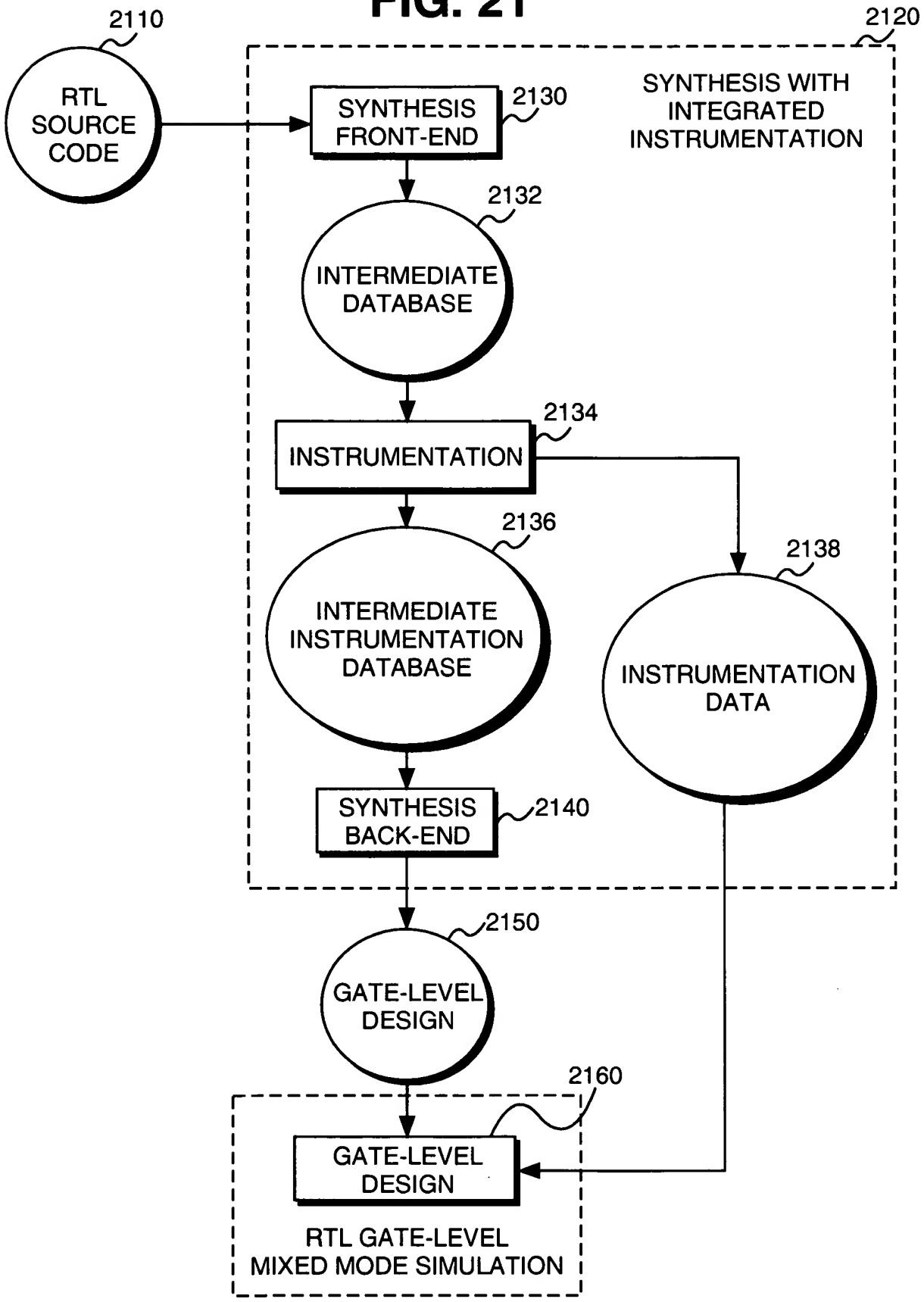
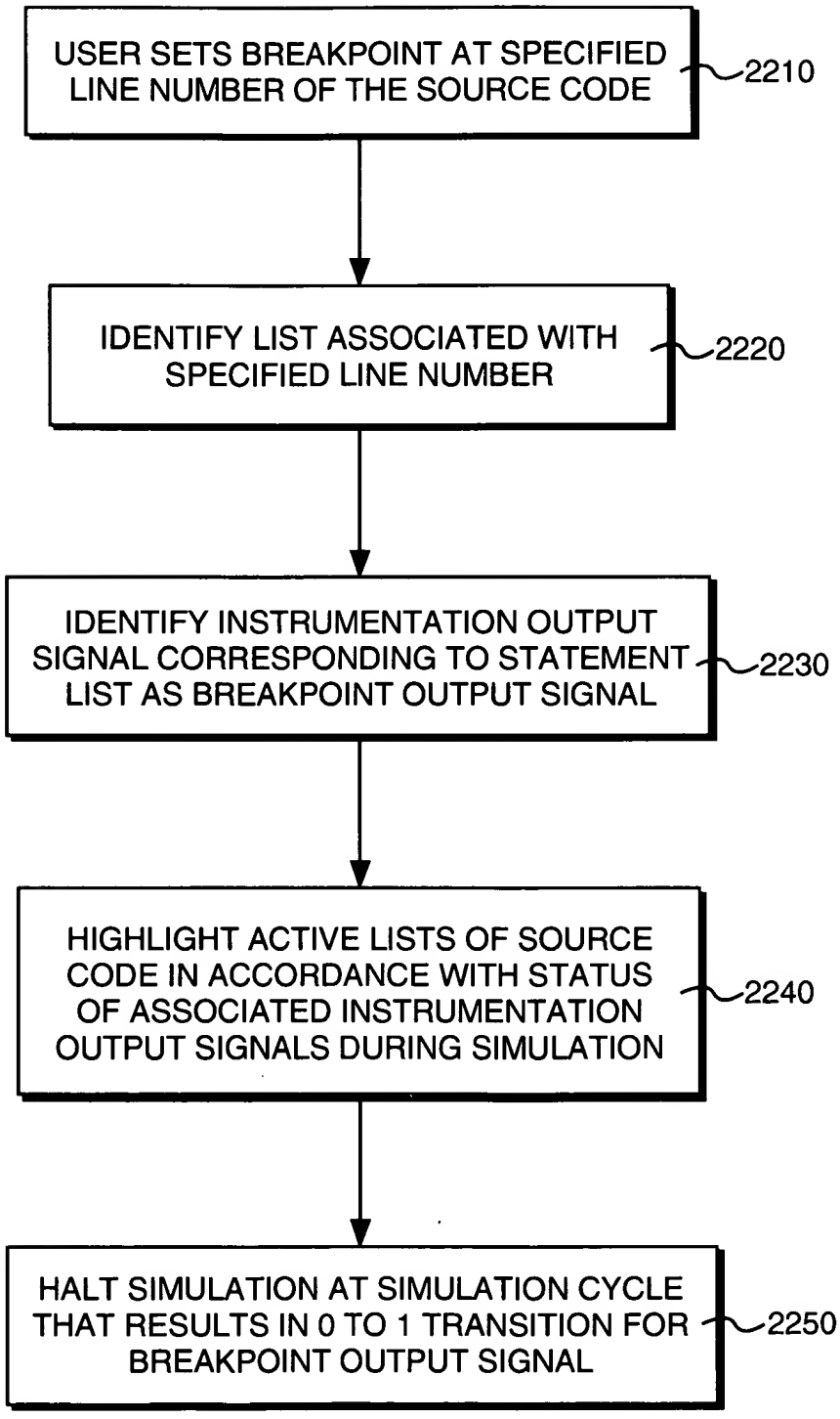

2110
RTL SOURCE CODE
2120
SYNTHESIS WITH INTEGRATED INSTRUMENTATION
SYNTHESIS FRONT-END
2130
2132
INTERMEDIATE DATABASE
INSTRUMENTATION
2134
2136
INTERMEDIATE INSTRUMENTATION DATABASE
2138
INSTRUMENTATION DATA
SYNTHESIS BACK-END
2140
2150
GATE-LEVEL DESIGN
2160
GATE-LEVEL DESIGN
RTL GATE-LEVEL MIXED MODE SIMULATION

FIG. 22


USER SETS BREAKPOINT AT SPECIFIED LINE NUMBER OF THE SOURCE CODE
2210
IDENTIFY LIST ASSOCIATED WITH SPECIFIED LINE NUMBER
2220
IDENTIFY INSTRUMENTATION OUTPUT SIGNAL CORRESPONDING TO STATEMENT LIST AS BREAKPOINT OUTPUT SIGNAL
2230
HIGHLIGHT ACTIVE LISTS OF SOURCE CODE IN ACCORDANCE WITH STATUS OF ASSOCIATED INSTRUMENTATION OUTPUT SIGNALS DURING SIMULATION
2240
HALT SIMULATION AT SIMULATION CYCLE THAT RESULTS IN 0 TO 1 TRANSITION FOR BREAKPOINT OUTPUT SIGNAL
2250